Artificial Intelligence for Software Architecture: Literature Review and the Road Ahead

ALESSIO BUCAIONI, Mälardalen Universty, Sweden
MARTIN WEYSSOW, JUNDA HE, YUNBO LYU, and DAVID LO, Singapore Management University, Singapore

This paper presents a forward-looking vision for artificial intelligence-driven software architecture that addresses longstanding challenges in design and evolution. Although artificial intelligence has achieved notable success in software engineering, its explicit application to software architecture remains under-explored. Traditional practices, heavily reliant on expert knowledge and complex trade-off reasoning, tend to be manual and error-prone, thereby compromising system quality and maintainability. Building on recent advances, we examine how artificial intelligence can automate architectural design, support quantitative trade-off analyses, and continuously update architectural documentation. Our approach combines a systematic review of state-of-the-art applications with insights from industry practitioners. The resulting roadmap outlines 14 current artificial intelligence contributions to software architecture, identifies six artificial intelligence-specific challenges in supporting architectural tasks, and reveals six avenues for future improvement, charting a course for future research and practical implementations.

CCS Concepts: • **Computer systems organization → Architectures**.

Additional Key Words and Phrases: Software architecture, Artificial Intelligence

ACM Reference Format:
Alessio Bucaioni, Martin Weyssow, Junda He, Yunbo Lyu, and David Lo. 2018. Artificial Intelligence for Software Architecture: Literature Review and the Road Ahead. In *Proceedings of The ACM International Conference on the Foundations of Software Engineering (FSE '25).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The profound impact of Artificial Intelligence (AI) on software engineering has been widely demonstrated in areas such as code generation, bug detection, and program repair. Despite this progress, the explicit application of AI to Software Architecture (SA) and its associated tasks remains relatively under-explored. SA is the blueprint upon which robust, scalable, and secure software systems are built [3]. Yet, the practice of designing and evolving SA is still largely manual, demanding extensive expertise, deep domain knowledge, and sophisticated reasoning about trade-offs [5]. Architects must continually analyse system constraints, anticipate evolution paths, and make decisions that balance different qualities such as performance, security, and reliability—challenges that require significant experience and effort. In many business contexts, the importance of SA is often overlooked in the rush to meet pressing time-to-market demands [20]. This neglect is concerning given that a well-designed architecture is critical to ensuring the long-term quality and maintainability of software systems [3].

Authors' Contact Information: Alessio Bucaioni, alessio.bucaioni@mdu.se, Mälardalen Universty, Sweden; Martin Weyssow, mweyssow@smu.edu.sg; Junda He, jundahe@smu.edu.sg; Yunbo Lyu, yunbolyu@smu.edu.sg; David Lo, davidlo@smu.edu.sg, Singapore Management University, Singapore.

The emerging promise of AI presents an opportunity to address these challenges by automating aspects of the architectural process and reducing the cognitive burden on architects. For example, recent advances suggest that AI can automatically derive partial design solutions from requirements [P2], perform quantitative trade-off analyses [P21], and even summarize and prioritize quality requirements [P27, P35]. Yet, integrating AI into the domain of SA is itself a formidable challenge. Unlike lower-level programming tasks, architectural design involves high-level planning, abstraction, and reasoning over long-term system evolution—areas where AI's current capabilities may struggle. This gap underscores a pivotal question: *how can AI be effectively used to overcome the inherent difficulties of architectural tasks without oversimplifying the complexity that SA demands?* Addressing this question is not merely a matter of leveraging cutting-edge technology; it is an essential step for every individual, team, and organization involved in software development. Quoting Ozkaya [14], the move toward AI-enhanced architectural practices should be driven by a clear understanding of the challenges at hand, rather than by a fear of missing out on technological trends.

In this paper, we propose a forward-looking vision for AI-driven SA along with a corresponding roadmap. To build this vision, we first analyse the current application of AI in SA through a systematic review, which reveals 14 state-of-the-art practices and applications in architectural tasks. We then integrate open practical SA challenges—identified in our previous study through 32 practitioner interviews [17]—with our systematic findings to assess the extent of AI contributions to SA. Based on this analysis, we identify six AI-specific challenges in supporting architectural tasks. Finally, we articulate our vision, which not only demonstrates how artificial intelligence can alleviate the reluctance to engage in complex architectural tasks and enhance overall decision quality, but also provides a concrete roadmap with six distinct directions for advancing AI in SA (AI4SA)—a critical endeavour for the future of software engineering.

The remainder of this paper is as follows. Section 2 describes the research methodology used to build our forward-looking vision for AI4SA. Section 3 presents the synthesis of the key insights from our systematic review, while Section 4 assesses AI's contributions to SA, identifies persisting gaps, and delineates the our vision.

2 Research Methodology

We built our forward-looking vision for AI4SA using a two-part methodology. First, we systematically reviewed the current use of AI in SA to map out state-of-the-art practices, techniques, and reported outcomes. Second, we integrated the insights gained from this review with a set of open SA challenges identified in our previous work [17] to assess the extent of AI contributions to SA. Based on this analysis, we identify AI-specific challenges, and articulated our vision, which encompasses six distinct directions for advancing AI4SA. This integrated approach not only documented current practices, but also highlighted emerging gaps and opportunities, forming the basis of our proposed roadmap for future research.

We designed and conducted our systematic review by following the guidelines for performing secondary studies in software engineering by Kitchenham [12]. The SLR helps us identify the current state of the art and formulate our vision based on scientific findings. The process we followed consists of three phases: planning, conducting, and documenting. The main objectives of the planning phase were to identify the Research Goal (RG) and Research Questions (RQs) , and define the research protocol for carrying out the study systematically. A detailed research protocol was the primary output of this phase. In the conducting phase, we performed activities outlined in the research protocol, including search and selection, data extraction form definition, data extraction, and data analysis. The main objectives of the analysis phase were to analyse and document potential threats to validity and to record the study's results. To facilitate independent replication and verification, we provide a complete replication package [1] containing search and selection data, and the list of primary studies.

2.1 Research Goal

Following the Goal-Question-Metric (GQM) approach [2], we defined the research goal, which is presented in Table 1, and then refined the goal in the research questions.

Table 1. Research goal expressed using the GQM perspectives.

Purpose	Identify, and classify
Issue	techniques and applications
Object	of AI for SA
Viewpoint	from the point of view of researchers and practitioners.

2.2 Search and selection

Following the steps illustrated in Figure 1, we collected relevant research studies for our investigation. We started with an automatic search of four of the largest scientific databases and indexing systems in software engineering [12]: *IEEE Xplore Digital Library*, *ACM Digital Library*, *SCOPUS* and *Web of Science*. We selected the above sources due to their

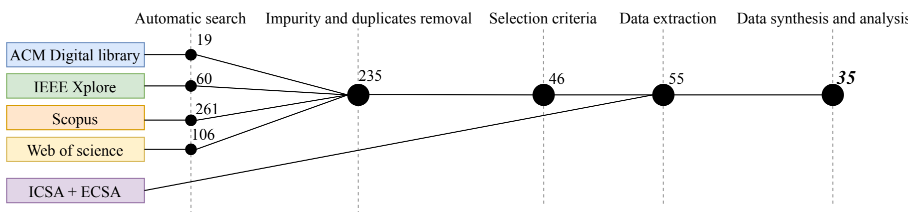

Fig. 1. Search and selection

recognized effectiveness in supporting systematic studies in software engineering [4]. We then queried these sources using the following search string on title and abstract:

("Artificial Intelligence" OR AI) AND "software architecture*"

We employed a relatively simple search string to gather as many potentially relevant studies as possible, resulting in an initial set of 446 studies. Note that we also applied a publication year filter, including only studies published from January 2019 to December 2024 (the time in which this review was conducted). We then removed non-research papers (impurities) and merged duplicates, resulting in a new set of 235 studies. Next, we applied the following inclusion (IC) and exclusion (EC) criteria based on the studies' titles, abstracts, and keywords:

- Inclusion criteria
  (1) Studies describing interplay between AI and SA.
- Exclusion criteria
  (1) Studies published as tutorial papers, short papers (less than 4 pages), poster papers, editorials and manuals.
  (2) Studies not available as full-text.
  (3) Secondary or tertiary studies.

[1] https://docs.google.com/spreadsheets/d/1oWElLL4EllSGZHBnrWasteBq4WmUAL3u/edit?usp=sharing&ouid=113010634542117145836&rtpof=true&sd=true

To proceed to the next stage, each study had to satisfy at least one inclusion criterion (IC) and none of the exclusion criteria (EC). This selection phase involved all authors in key decisions, yielding a set of 46 peer-reviewed studies. To mitigate potential construct validity biases [9] and ensure the inclusion of relevant studies from the SA community, we manually screened the proceedings (including companion proceedings) of the International Conference on Software Architecture (ICSA) and the European Conference on Software Architecture (ECSA) from 2019 to 2024, using the same selection criteria. This step added 9 more studies. Subsequently, we conducted closed recursive backward and forward snowballing [19], but identified no additional sources resulting in a total of 55 studies from the automatic search and selection process.

2.3 Data extraction, analysis and synthesis

To extract and collect data from the primary studies, we initially gathered all pertinent data from the primary studies and then applied the grounded theory methodology [6], which systematically breaks down data, categorizes it, and establishes connections between categories to uncover emerging themes. This approach enabled us to organize qualitative data in a structured way, thereby providing deeper insights into the interplay between AI and SA. During the data extraction, we further removed studies from which we could not extract any relevant information obtain a final set of 35 primary studies that are listed in the Primary Studies appendix. We analysed and synthesized the extracted data by following the guidelines of Cruzes et al. [7]. We began by examining each primary study individually, classifying key features according to the parameters specified in the data extraction form. Next, we analysed the entire set of primary studies to identify and interpret emerging patterns.

2.4 Integration of open software architecture challenges

Concurrently with the systematic literature review, all authors engaged in a qualitative process to revisit a comprehensive list of open SA challenges identified in the previous study of Wan et al. that interviewed 32 practitioners [17]. Drawing on our collective expertise in AI and SA research, we mapped our systematic review findings to these challenges, thereby clearly delineating both current capabilities and research gaps. This mapping directly informs our roadmap for advancing AI4SA.

2.5 Threats to validity

Regarding external validity, the selected papers may not fully capture all significant literature. To mitigate this risk, we ran our search in four major software engineering databases and supplemented these efforts with a manual screening of the proceedings (including companion proceedings) of ICSA and ECSA, as well as recursive backward and forward snowballing. Although the simplicity of our search string and inclusion/exclusion criteria might create the possibility of overlooking certain works, this choice was intentional to maintain broad coverage. To address internal validity, we followed established guidelines for systematic studies. We employed descriptive statistics, cross-analysed the categories in the extraction form, and performed sanity checks to ensure data consistency. Regarding construct validity, a poorly formulated search string could undermine our findings; however, we used a straightforward string requiring minimal adjustments to reduce the risk of missing critical studies. Finally, we safeguarded conclusion validity by consistently applying well-defined processes and by making a publicly accessible replication package available. All authors were involved in defining the extraction form and carried out the data extraction, analysis, and synthesis, relying on insights derived from our experience.

3 Review Synthesis: The Current State of Artificial Intelligence in Software Architecture

In this section, we present a synthesis of the key insights from our systematic review, revealing three major clusters that encapsulate the interplay between AI and SA. The first cluster, "AI for Architecture Design and Decision-Making," and the second, "AI for Architecture Evolution and Adaptation," demonstrate how AI techniques are being harnessed to enhance architectural design, support decision-making, and drive continuous system evolution. The third cluster, "Architecture for AI Systems," highlights how traditional SA principles and practices are evolving to accommodate the unique requirements of AI-based technologies. However, given the scope of our work and the limitations imposed by space, we focus our discussion on the first two clusters, which are summarized in Table 2

Table 2. AI and SA interplay

Contribution	Cluster	Topic	Study
AI for SA	C1-AI for Architecture Design and Decision-Making	T1-AI-Assisted Architecture Design	[P2, P17, P21, P22]
		T2-AI-Assisted Design Pattern Recognition	[P15]
		T3-AI-Assisted Architecture Decision Analysis	[P30]
		T4-AI-Assisted Decision Making	[P28, P32]
		T5-AI Knowledge Representation	[P7]
		T6-AI-assisted Design Decision	[P11]
	C2-AI for Architecture Evolution and Adaptation	T7-ML component adaptation	[P34]
		T8-AI-Assisted Architecture Recovery and Reverse Engineering	[P26, P31]
		T9-AI-Assisted Resource Management	[P25]
		T10-AI-Assisted Performance Estimation	[P12]
		T11-AI-Enabled Industrial Automation	[P8]
		T12-AI-Assisted Refactoring	[P5]
		T13-AI-Assisted Resilient Architecture	[P4]
		T14-AI-Assisted QoS	[P27, P28, P35]

AI for Architecture Design and Decision-Making. A growing body of work investigates how AI can help architects make more informed or automated decisions. Research on AI-assisted architecture design has reported: (i) the use of LLMs for generating SA or pattern candidates based on requirements [P2, P17, P22], (ii) natural language processing (NLP), graph analysis, and community detection (including AI-based language models) for naming microservices [P21], and (iii) LLM-based quantitative analyses for automatically performing trade-off analysis of the generated architecture candidates [P2]. In the area of AI-Assisted Design Pattern Recognition, studies examined language models based on BERT to determine their effectiveness in recognizing the Singleton design pattern [P15]. For AI-Assisted Architecture Decision Analysis, knowledge graphs were employed to support the analysis of Architecture Decision Records (ADRs)[P30]. Meanwhile, work in AI-Assisted Decision Making used machine learning (ML) for proactive decision-making and for representing decision makers in product line design optimization[P28, P32]. Lastly, studies on AI-Assisted Design Decision leveraged LLMs to generate architectural design decisions [P11].

AI for Architecture Evolution and Adaptation. AI-based methods can dynamically reconfigure system components to enhance qualities such as interoperability, resilience, and continuous optimization. This second cluster encompasses several interrelated topics. Research on ML component adaptation introduces estimators to simplify the use of ML components in self-adaptive architectures [P34]. Work on AI-Assisted Architecture Recovery and Reverse Engineering investigated AI techniques to extract and analyze SA from codebases [P26], as well as recommender systems for recovering relevant JavaScript packages from web repositories [P31]. In AI-Assisted Resource Management, centralized AI-based approaches have been proposed to handle resource allocation more effectively [P25]. AI-Assisted Performance Estimation explored how to measure potential performance gains when AI jobs are executed on different SAs [P12]. The field of AI-Enabled Industrial Automation highlighted hyperautomation, which integrates event-driven software architecture with AI and ML to streamline industrial processes [P8]. AI-Assisted Refactoring demonstrated how AI can support the restructuring of large monolithic applications [P5], and AI-Assisted Resilient Architecture enhanced fault tolerance in back-end services [P4]. Finally, AI-Assisted QoS research applied reinforcement learning to achieve interoperability in IoT applications [P27] and used ML to improve the QoS of microservices architectures [P35].

4 The Road Ahead: Charting a Vision for Artificial Intelligence-Driven Software Architecture

Despite AI's growing role in SA, its current capabilities still fall short of addressing key challenges. As Ozkaya suggests [14], the shift toward AI-enhanced architectural practices should be guided by a clear understanding of these challenges rather than a fear of missing out. In Section 4.1, we align our systematic findings with 17 open SA challenges—identified through 32 practitioner interviews [17]—to pinpoint AI-specific challenges in architectural tasks. Section 4.2 then outlines our vision for overcoming these challenges by detailing six distinct directions for advancing AI-driven software architecture.

4.1 Challenges in Artificial Intelligence-Driven Software Architecture

Table 3 summarizes the 17 open software architecture challenges (SACH) from our previous study [17], detailing how current AI practices (T) address these arhcitectural challenges and highlighting the persistent AI-specific challenges (AICH).

Table 3. Software architecture challenges highlighted in Wan et al.'s study [17] and their mapping with AI for SA topics highlighted in our literature review.

SA challenge	AI for SA Topic	AI-specific challenge
SACH1-Unpredictable Evolution and Changes of Software Requirements	T1, T3, T4	AICH1- Move beyond one-time recommendations
SACH2-Architecture Documentation Becomes Obsolete as Software Evolves	T5, T8	AICH2-Ensure traceability and alignment
SACH3-Unclear Boundaries Between Architectural Elements	T1, T2	AICH3-Develop context-aware reasoning
SACH4-Interdisciplinary Knowledge Required to Lower Coupling and Improve Cohesion	T1	
SACH5-Architecture Review Requires a Standard Process, External Expertise, and Tool Support	T3, T4	AICH4-Incorporate domain expertise
SACH6-Lack of Effective and Apply-to-All Quantitative Measures	T10, T14	AICH5- Learn from past architectural decisions and dynamically refine decisions
SACH7-Automated Architecture Conformance Checking Is Rare	T8	
SACH8-Obsolete Documentation and Lack of Traceability Hinder Conformance Checking	T8	
SACH9-Limited Tool Support to Continuously Monitor the Health of Software Architectures	T9, T10, T13, T14	
SACH10-Pinpointing Architecture Problems Requires a System-wide Perspective	T8, T12	AICH6- Integrates long-term architectural evolution and technical debt reduction into recommendations.
SACH11-Technical Debts Are Introduced to Software Projects	T12, T8	
SACH12-Lack of Tool Support for Detecting Architectural Smells	T8, T12	
SACH13-Unawareness of Correlation Between Code Smells and Architecture Problems	T12, T8	
SACH14-Lack of Tool Support to Capture and Aggregate Symptoms of Architecture Erosion	T12, T8	
SACH15-No Agreement on the Value of Architecture Refactoring	T12, T10	
SACH16-Inadequate Tool Support for Impact Analysis of Architectural Changes	T3, T4	
SACH17-Inadequate Tool Support for Module- and System-Level Refactoring	T12	

Software systems evolve continuously, yet volatile requirements (SACH1) make their evolution uncertain. While AI generates alternative designs and analyzes trade-offs using historical data [P2, P15, P17, P21, P22, P28, P32], these methods are reactive and cannot adapt to unforeseen changes. For instance, LLM-based approaches [P2] generate architectures from requirements but require full re-runs when updates occur, and models fine-tuned on static datasets [P17] cannot adjust without retraining. The AI-specific challenge (AICH1) is to move beyond one-time recommendations and support continuous architectural evolution.

Similarly, The challenge of keeping architectural documentation current (SACH2) is only partly addressed by AI-based approaches such as architecture recovery and reverse engineering [P31, P26, P17, P2, P11]. These methods mainly extract structural elements rather than maintain semantic documentation reflecting architectural rationale. For example, while [P11] uses language models to generate architectural decisions, these are not integrated into living documentation, leaving architects to manually update records. Here, the AI-specific challenge (AICH2) is to ensure traceability and alignment between evolving software and its documentation.

Identifying architectural structures and boundaries (SACH3) has improved using statistical methods and language models [P15, P17, P21, P22], yet these approaches often lack the semantic precision for complex decompositions. For example, [P15] targets specific design patterns like Singleton, while [P17] predicts high-level styles without fine-grained detail. Similarly, maintaining loosely coupled, cohesive systems (SACH4) remains challenging, as current AI tools offer isolated suggestions rather than ensuring system-wide modularity. The AI-specific challenge (AICH3) is to advance beyond pattern matching and develop context-aware reasoning about dependencies and architectural decomposition.

In architecture review (CH5), AI can suggest improvements and integrate diverse data sources [P2, P28, P30], yet it often misses the nuanced insights of experienced architects, leading to explainability and trust issues. For example, while knowledge graphs and language models aid decision-making [P2, P30], they lack a comprehensive evaluation framework. The AI-specific challenge (AICH4) is to incorporate domain expertise and interpretable reasoning to foster trust and adoption.

AI-assisted performance estimation and quality-of-service measures automate parts of software architecture assessment but lack universal, quantitative metrics (SACH6), as they are often domain-specific. Similarly, automated conformance checking (SACH7, SACH8) remains incomplete due to limited contextual understanding, affecting continuous health monitoring (SACH9). The AI-specific challenge (AICH5) is for AI to learn from past architectural decisions and dynamically refine evaluation metrics, optimizing trade-offs in performance, maintainability, security, and cost.

The absence of unified metrics complicates pinpointing architectural problems (SACH10), as AI's module-level diagnostics [P5, P26, P30, P31] fall short of a comprehensive, system-wide tool. Similarly, its capacity to detect architectural smells and mitigate technical debt (SACH11–CH14) is constrained by the diversity of smells and context-sensitive thresholds. Moreover, while AI can suggest local refactoring (SACH15) [P5, P12], it fails to consider long-term maintainability trade-offs (SACH17). The AI-specific challenge (AICH6) is to evolve into a system-aware diagnostic and refactoring assistant that integrates long-term architectural evolution and technical debt reduction into its recommendations.

4.2 A Vision for Artificial Intelligence-DrivenHello Software Architecture

Our analysis identifies six key AI-specific challenges in software architecture (AICH1–AICH6). To address these, we propose a forward-looking vision for AI-driven software architecture structured around the following dimensions.

Real-Time Monitoring and Self-Adaptation (AICH1). We envision AI-driven SA tools that continuously monitor architecture health, evolving software requirements, and performance degradation, determining when architectural changes are needed. AI will anticipate shifts, autonomously refine architectures, and guide self-adaptive decision-making to sustain long-term system health. To achieve this, LLM-based agents with tool integration will leverage external monitoring systems and runtime analytics to gather real-time insights on software evolution. By incorporating dynamic contextual information, AI will update its recommendations over time, refining the SA as the system and requirements evolve.

Automated Documentation and Traceability (AICH2). . To reduce the manual burden on architects, AI should continuously extract knowledge from codebases, architectural decisions, and other artifacts to ensure documentation remains accurate and aligned with system changes. LLMs can suggest updates based on real-time system insights, which helps architects maintain relevant documentation with minimal manual intervention. AI-driven traceability mechanisms will link architectural decisions to their implementation. By embedding AI-driven documentation into the development workflow, we shift from static records to a dynamic, continuously updated knowledge base, which helps prevent architectural drift and mitigates the accumulation of technical debt.

Context-Aware and Explainable AI (AICH3-4). . We envision next-generation AI-driven SA tools that move beyond static pattern matching to develop deep contextual reasoning and provide transparent, interpretable recommendations that architects can rely on. To achieve this, LLM-based approaches must incorporate domain-specific adaptation, either through fine-tuning on specialized data or retrieval-augmented generation (RAG) to dynamically integrate architectural knowledge. Additionally, graph-based AI models (e.g., graph neural networks) can provide structural representations of SA, allowing AI to capture interdependencies across software components at multiple levels of abstraction. Beyond generating recommendations, AI must justify its decisions. Reasoning-driven LLMs (e.g., OpenAI o1 [8], DeepSeek-R1 [10]) will provide step-by-step justifications for architectural decisions, making AI-driven insights more transparent and actionable. Interactive planning mechanisms [15, 16, 18] will enable architects to refine AI-generated recommendations iteratively in a collaborative decision-making process. Finally, Reinforcement Learning with Human Feedback (RLHF) [1, 13] will enhance AI-driven SA tools by aligning LLM recommendations with architect preferences. By learning from expert feedback, LLMs can refine their reasoning while improving both the accuracy and relevance of their suggestions.

Multi-Objective Optimization (AICH5). A future vision involves models that can intelligently balance competing architectural trade-offs, such as performance, maintainability, security, and cost. Multi-objective reinforcement learning [11, 21] offers a powerful framework for optimizing across these conflicting objectives when recommending architectural candidates or refining existing SA. By incorporating multi-objective optimization, AI can evolve from a rigid decision-support tool into a flexible, adaptive collaborator, allowing architects to dynamically explore trade-offs and make context-aware design decisions that adapt throughout the software lifecycle.

Integrated Multi-Level Diagnostics (AICH6). We envision AI-driven SA tools that provide holistic system analysis, identifying and correlating issues across multiple abstraction levels, such as linking low-level code smells to high-level architectural flaws. Future AI-driven approaches will integrate graph-based models to correlate issues across system layers, enabling intelligent diagnostics that dynamically adapt to architectural evolution.

Benchmarks and Industrial Studies. A forward-looking vision for AI-driven SA should include the development of high-quality, realistic, and representative benchmarks for training and evaluating AI solutions in automating tasks traditionally performed by software architects. One promising approach is synthetic data generation, which eases the tedious task of collecting high-quality architectural data. Equally important is evaluating AI solutions in real-world settings by integrating them into architects' daily workflows—such as through AI plug-ins for popular development tools. This dual focus on robust benchmarking and practical evaluation is crucial for advancing the field and ensuring that AI systems meet software architects' needs.

5 Conclusion

In this paper, we propose a forward-looking vision for AI-driven SA. Our systematic review identified 14 state-of-the-art practices, and by integrating practical challenges, we revealed six AI-specific challenges. From this analysis, we derived six directions for advancing AI in SA—a critical step toward automating complex tasks and enhancing the engineering process.

Primary studies

[P1] Philipp Haindl, Georg Buchgeher, Maqbool Khan, and Bernhard Moser. 2022. Towards a reference software architecture for human-ai teaming in smart manufacturing. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*. 96–100.

[P2] Tobias Eisenreich, Sandro Speth, and Stefan Wagner. 2024. From requirements to architecture: An ai-based journey to semi-automatically generate software architectures. In *Proceedings of the 1st International Workshop on Designing Software*. 52–55.

[P3] Fabrizio Nunnari and Daniel Sonntag. 2021. A software toolbox for deploying deep learning decision support systems with XAI capabilities. In *Companion of the 2021 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. 44–49.

[P4] Lorenzo Barberis Canonico, Vimal Vakeel, James Dominic, Paige Rodeghero, and Nathan McNeese. 2020. Human-AI partnerships for chaos engineering. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 499–503.

[P5] Gargi B Dasgupta. 2021. AI and its Applications in the Cloud strategy. In *Proceedings of the 14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference)*. 1–1.

[P6] Anna Lekova, Paulina Tsvetkova, and Anna Andreeva. 2023. System software architecture for enhancing human-robot interaction by conversational ai. In *2023 International Conference on Information Technologies (InfoTech)*. IEEE, 1–6.

[P7] Jasmin Jahić, Robin Roitsch, and Łukasz Grzymkowski. 2021. Knowledge-based Adequacy assessment Approach to support AI adoption. In *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 8–14.

[P8] D Sudharson, A Bhuvaneshwaran, TR Kalaiarasan, V Sushmita, et al. 2023. A multimodal AI framework for hyper automation in industry 5.0. In *2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA)*. IEEE, 282–286.

[P9)] Qinghua Lu, Liming Zhu, Xiwei Xu, Yue Liu, Zhenchang Xing, and Jon Whittle. 2024. A taxonomy of foundation model based systems through the lens of software architecture. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. 1–6.

[P10] Leonhard Faubel, Thomas Woudsma, Leila Methnani, Amir Ghorbani Ghezeljhemeidan, Fabian Buelow, Klaus Schmid, Willem D Van Driel, Benjamin Kloepper, Andreas Theodorou, Mohsen Nosratinia, et al. 2024. A MLOps Architecture for XAI in Industrial Applications. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 1–4.

[P11] Rudra Dhar, Karthik Vaidhyanathan, and Vasudeva Varma. 2024. Can LLMs Generate Architectural Design Decisions?-An Exploratory Empirical study. *arXiv preprint arXiv:2403.01709* (2024).

[P12] Audrey Berquand, Francesco Murdaca, Annalisa Riccardi, Tiago Soares, Sam Generé, Norbert Brauer, and Kartik Kumar. 2019. Artificial intelligence for the early design phases of space missions. In *2019 IEEE Aerospace Conference*. IEEE, 1–20.

[P13] Zhicheng Bao, Weishan Zhang, Xingjie Zeng, Hongwei Zhao, Cihao Dong, Yuming Nie, Yuru Liu, Yuange Liu, and Junzhong Wu. 2023. Software Architecture for Responsible Artificial Intelligence Systems: Practice in the Digitization of Industrial Drawings. *Computer* 56, 4 (2023), 38–49.

[P14] Ran Bi, Tongtong Xu, Mingxue Xu, and Enhong Chen. 2022. PaddlePaddle: A Production-Oriented Deep Learning Platform Facilitating the Competency of Enterprises. In *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 92–99.

[P15] Sushant Kumar Pandey, Miroslaw Staron, Jennifer Horkoff, Miroslaw Ochodek, and Darko Durisic. 2024. Comparing Programming Language Models for Design Pattern Recognition. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 183–190.

[P16)] Ammar Yasser and Mervat Abu-Elkhier. 2021. Towards fluid software architectures: bidirectional human-AI interaction. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 1368–1372.

[P17] Brandon Gustrowsky, Joel López Villarreal, and Germán H Alférez. 2024. Using Generative Artificial Intelligence for Suggesting Software Architecture Patterns from Requirements. In *Intelligent Systems Conference*. Springer, 274–283.

[P18] Evangelos Ntentos, Stephen John Warnett, and Uwe Zdun. 2024. Supporting Architectural Decision Making on Training Strategies in Reinforcement Learning Architectures. In *2024 IEEE 21st International Conference on Software Architecture (ICSA)*. IEEE, 90–100.

[P19] Hamed Sahebi and Dan Sui. 2024. Develop AI-Aid Drilling Environment via Micro Service Systems. In *International Conference on Offshore Mechanics and Arctic Engineering*, Vol. 87868. American Society of Mechanical Engineers, V008T11A061.

[P20] Francisco Pinto-Santos, Juan Antonio González-Ramos, Raúl López-Blanco, and Javier Curto. 2023. A New Software Platform to Provide AI-Based Services to Smart Cities. In *International Symposium on Ambient Intelligence*. Springer, 84–92.

[P21] Sebastian Arias, Aracely Suquisupa, Maria Fernanda Granda, and Víctor Saquicela. 2024. Generation of Microservice Names from Functional Requirements: An Automated Approach. In *New Perspectives in Software Engineering*. Springer, 157–173.

[P22] Guillermo Rodriguez, J Andrés Díaz-Pace, Luis Berdun, and Sanjay Misra. 2021. Deriving Architectural Responsibilities from Textual Requirements. In *International Conference on Informatics and Intelligent Applications*. Springer, 297–310.

[P23] Antonio Martínez-Rojas, J Sánchez-Oliva, José Manuel López-Carnicer, and Andrés Jiménez-Ramírez. 2021. Airpa: An architecture to support the execution and maintenance of AI-powered RPA robots. In *International Conference on Business Process Management*. Springer, 38–48.

[P24] Li Li, Jun Wang, and ChengZhong Xu. 2020. Flsim: An extensible and reusable simulation framework for federated learning. In *International Conference on Simulation Tools and Techniques*. Springer, 350–369.

[P25] Lasal Sandeepa Hettiarachchi, Senura Vihan Jayadeva, Rusiru Abhisheak Vikum Bandara, Dilmi Palliyaguruge, Udara Srimath S Samaratunge Arachchillage, and Dharshana Kasthurirathna. 2022. Artificial Intelligence-Based Centralized Resource Management Application for Distributed Systems. In *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 1–6.

[P26] Dragoş Dobrean. 2019. Automatic examining of software architectures on mobile applications codebases. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 595–599.

[P27] Sebastian Kotstein and Christian Decker. 2019. Reinforcement learning for IoT interoperability. In *2019 IEEE international conference on software architecture companion (ICSA-C)*. IEEE, 11–18.

[P28] Henry Muccini and Karthik Vaidhyanathan. 2019. A machine learning-driven approach for proactive decision making in adaptive architectures. In *2019 IEEE international conference on software architecture companion (ICSA-C)*. IEEE, 242–245.

[P29] Haruki Yokoyama. 2019. Machine learning system architectural pattern for improving operational stability. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 267–274.

[P30] Aleksei Karetnikov, Lisa Ehrlinger, Georg Buchgeher, and Verena Geist. 2024. Semantic Modeling of Architecture Decision Records to Enable AI-based Analysis. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 62–66.

[P31] Hernan C Vazquez, J Andres Diaz-Pace, Santiago A Vidal, and Claudia Marcos. 2023. A recommender system for recovering relevant javascript packages from web repositories. In *2023 IEEE 20th International Conference on Software Architecture (ICSA)*. IEEE, 175–185.

[P32] Fernando H Kuviatkovski, Willian M Freire, Aline MMM Amaral, Thelma E Colanzi, and Valéria D Feltrim. 2022. Evaluating machine learning algorithms in representing decision makers in search-based PLA. In *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 68–75.

[P33] Qinghua Lu, Liming Zhu, Xiwei Xu, Zhenchang Xing, Stefan Harrer, and Jon Whittle. 2024. Towards responsible generative ai: A reference architecture for designing foundation model based agents. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 119–126.

[P34] Milad Abdullah, Michal Töpfer, Tomáš Bureš, Petr Hnětynka, Martin Kruliš, and František Plášil. 2022. Introducing Estimators—Abstraction for Easy ML Employment in Self-adaptive Architectures. In *European Conference on Software Architecture*. Springer, 370–385.

[P35] Neha Kaushik. 2024. Improving QoS of Microservices Architecture Using Machine Learning Techniques. In *European Conference on Software Architecture*. Springer, 72–79.

References

[1] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862* (2022).

[2] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. 1994. The Goal Question Metric Approach. In *Encyclopedia of Software Engineering*. Vol. 2. Wiley, 528–532.

[3] Len Bass. 2012. *Software architecture in practice*. Pearson Education India.

[4] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software* (2007).

[5] Alessio Bucaioni, Amleto Di Salle, Ludovico Iovino, Patrizio Pelliccione, and Franco Raimondi. 2024. Architecture as Code. In *22nd IEEE International Conference on Software Architecture*.

[6] Kathy Charmaz and Linda Liska Belgrave. 2007. Grounded theory. *The Blackwell encyclopedia of sociology* (2007).

[7] Daniela S Cruzes and Tore Dyba. 2011. Recommended steps for thematic synthesis in software engineering. In *Procs of ESEM*.

[8] Tianyang Zhong et al. 2024. Evaluation of OpenAI o1: Opportunities and Challenges of AGI. arXiv:2409.18486 [cs.CL] https://arxiv.org/abs/2409.18486

[9] Trisha Greenhalgh and Richard Peacock. 2005. Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. *BMJ* 331, 7524 (2005), 1064–1065.

[10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[11] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. 2022. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems* 36, 1 (2022), 26.

[12] Barbara Kitchenham and Pearl Brereton. 2013. A systematic review of systematic review process research in software engineering. *Information and software technology* (2013).

[13] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. (2023).

[14] Ipek Ozkaya. 2023. The next frontier in software development: AI-augmented software development processes. *IEEE Software* 40, 4 (2023), 4–9.

[15] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813* (2023).

[16] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems* 36 (2023), 75993–76005.

[17] Zhiyuan Wan, Yun Zhang, Xin Xia, Yi Jiang, and David Lo. 2023. Software architecture in practice: Challenges and opportunities. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1457–1469.

[18] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560* (2023).

[19] Claes Wohlin. 2014. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In *Procs of EASE*. ACM, Article 38, 10 pages. doi:10.1145/2601248.2601268

[20] Lu Xiao, Yuanfang Cai, Rick Kazman, Ran Mo, and Qiong Feng. 2016. Identifying and quantifying architectural debt. In *Proceedings of the 38th international conference on software engineering*. 488–498.

[21] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in neural information processing systems* 32 (2019).

Received February 2025; revised ; accepted